جامعة طرابلس ـ كلية تقنية المعلومات

Design and Analysis Algorithms

تصميم و تحليل خوارزميات

ITGS301

Lecture 10 : المحاضرة العاشرة

Tripoli University - Faculty of Information Technology
Marwa Solla
m.solla@uot.edu.ly

Graph Algorithms

What is a Graph?

A Graph is a abstract data structure represents a collection of items with pairwise relationship between these items.

It consists of a set of vertices (nodes) connected by a set of edges (links), and is denoted by G = (V, E), where V is set of vertices and E is set of edges.

- *V(G)* is a set of vertices or nodes which can represent an object that needs to be “connected”.

- *V* represents the number of vertices (nodes) in the graph

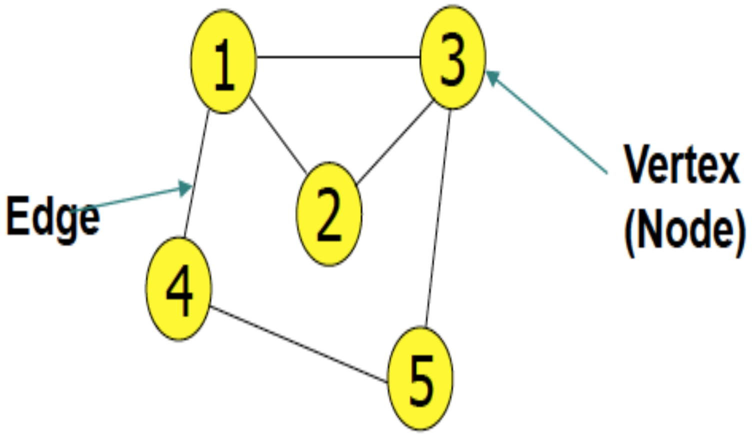
- *E(G)* is a set of edges. An edge is a distinct pair of vertices. An edge indicates a valid/existing connection between two vertices.

- *E* represents the number of edges in the graph

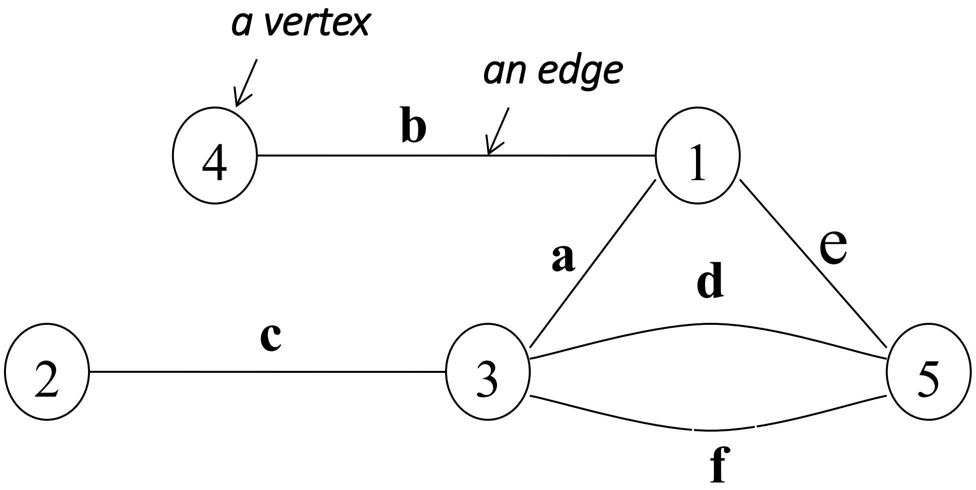
$G = (V, E)$

V = set of vertices $|V| = n$

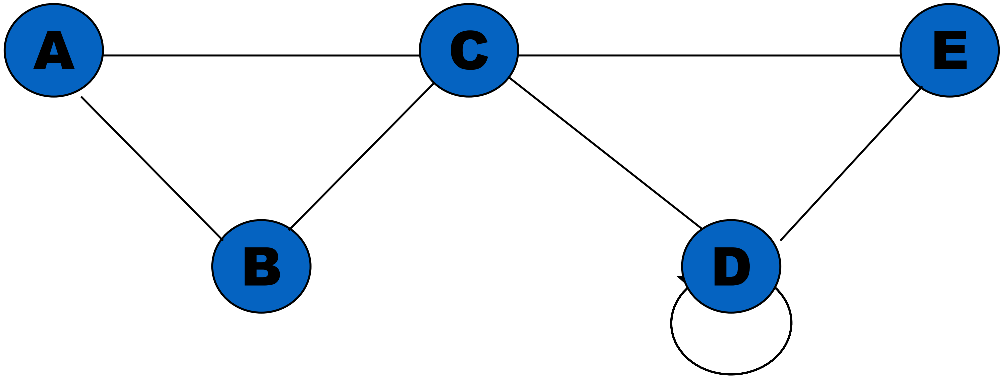
E = set of edges $|E| = m$



An example of a graph



Another Example



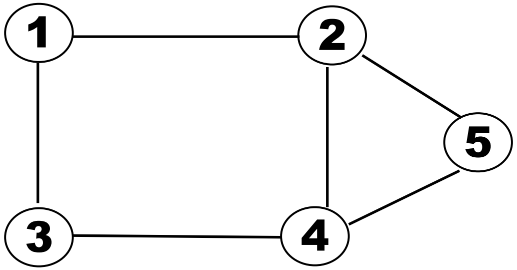
V = 5

V(G) = {A,B,C,D,E}

E(G) = {(AC), (AB), (BC), (CD), (CE), (DE), (DD)}

Graph representation

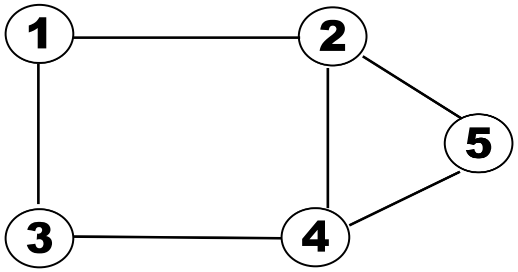
1. **Adjacency Matrix:** represent a graph as n*n Matrix A



	1	2	3	4	5
1	0	1	1	0	0
2	1	0	0	1	1
3	1	0	0	1	0
4	0	1	1	0	1
5	0	1	0	1	0

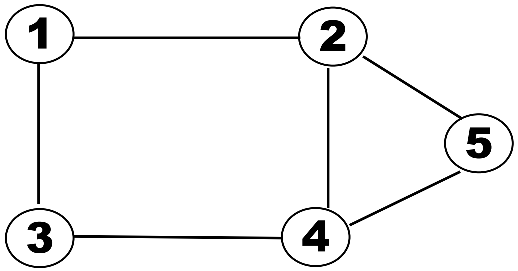
Graph representation

1. Adjacency Matrix: represent a graph as n*n Matrix A



	1	2	3	4	5
1	0	1	1	0	0
2	1	0	0	1	1
3	1	0	0	1	0
4	0	1	1	0	1
5	0	1	0	1	0

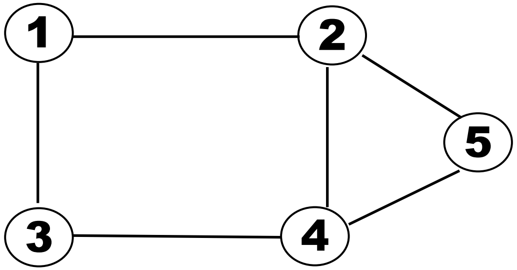
Graph representation

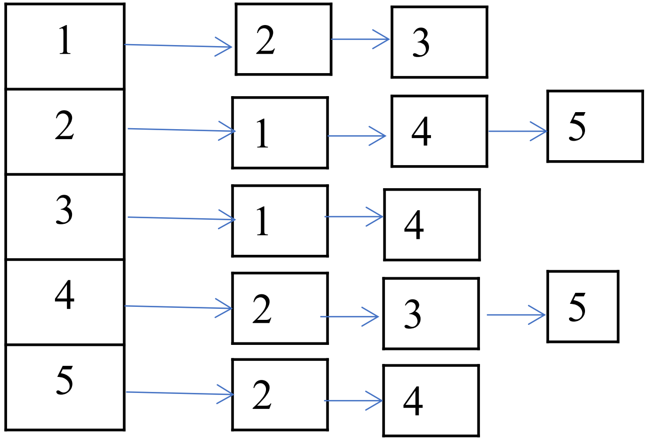


	1	2	3	4	5
1	0	1	1	1	0
2	1	0	1	0	1
3	1	1	0	1	1
4	1	0	1	0	1
5	0	1	1	1	0

Graph representation

2. Adjacent List: for each vertex v ∈ V , store a list of vertices adjacent to v





1 → 2 → 3

2 → 1 → 4 → 5

3 → 1 → 4

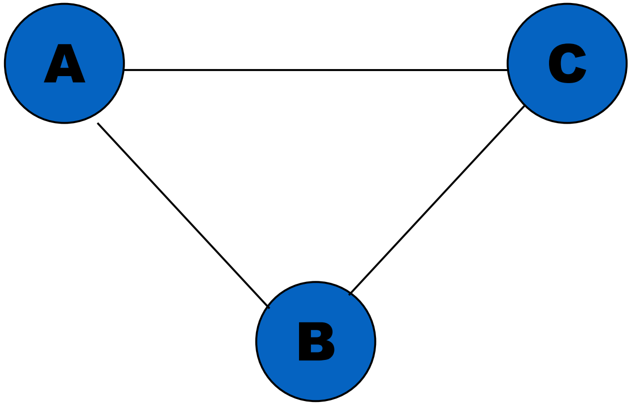
4 → 2 → 3 → 5

5 → 2 → 4

Graph Terminology

- Adjacent Vertices

if two vertices are joined by an edge they are said to be ***adjacent***



Adjacent Vertices:

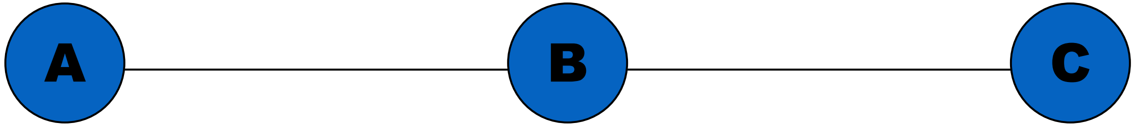
A & C

A & B

B & C

- **Degree**

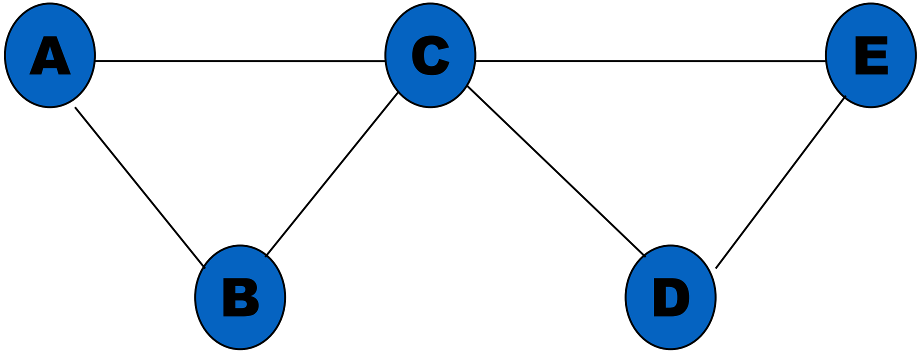
  - the degree if a vertex ***x*** is the number of vertices adjacent to it (or the number of edges incident to it)
  - represented as ***deg(x)***



deg(A) = 1　*deg(B) = 2*　*deg(C) = 1*

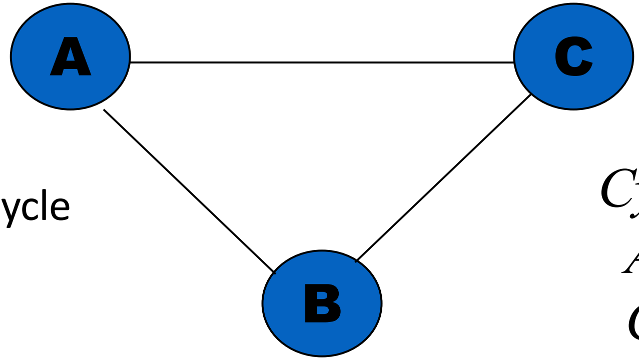
- **Path**

  – a path is sequence of vertices in which each vertex is adjacent to the next one.



Path from A to D: A-B-C-D
Path from B to E: B-A-C-D-E

- **Cyclic**
  - cycle is a path consisting of *at least three vertices* that started and ends with the same vertex.
  - So the graph is a cycle if there is subgraph is cycle.

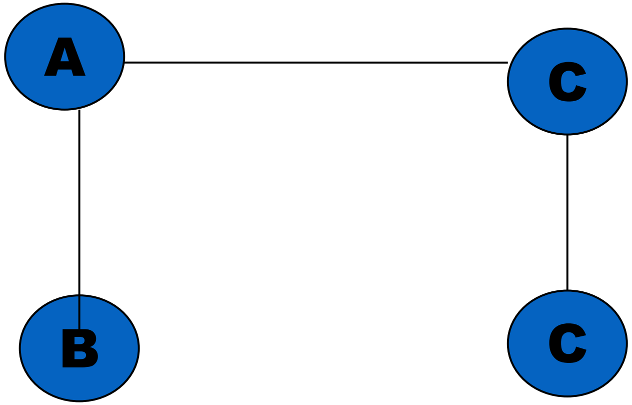


ABC is a cycle

Cycles:
A-B-C-A
C-B-A-C

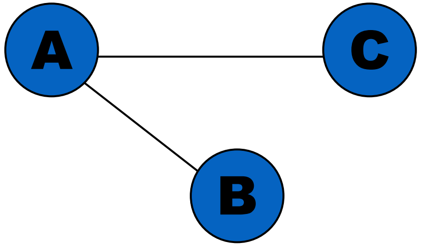
Acyclic

A graph is acyclic if no subgraph is a cycle>

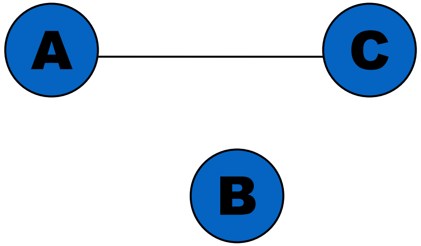


• Connected

– a graph ***G*** is connected if there is at least one path from every vertex to every other vertex in the graph .



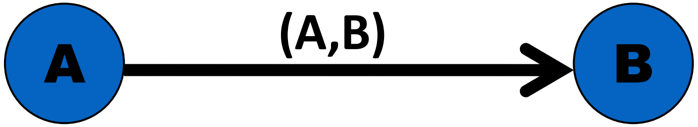
Connected Graph



Disconnected Graph

Types of Graph

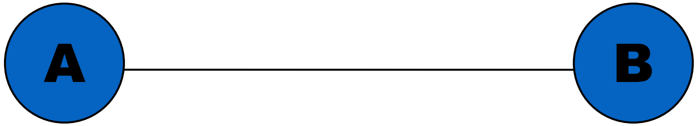
- Directed Graph or Digraph
  - the connecting lines are usually represented with an arrow



Note: (A,B) ⍰ (B,A)

• Undirected Graph

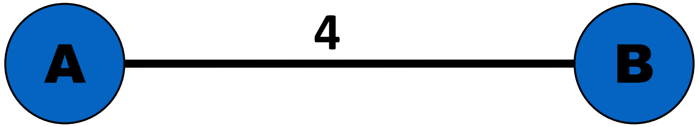
- the order of the vertices in the pair of vertices in the set of edges does not matter



(A,B) = (B,A)

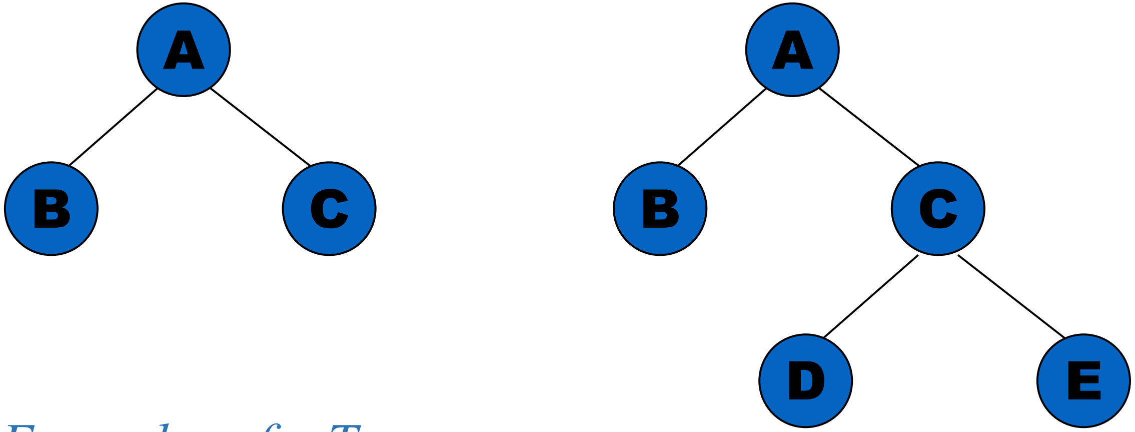
• Weighed Graph

- each edge has an associated weight which could indicate cost, distance, time, etc. between two adjacent vertices

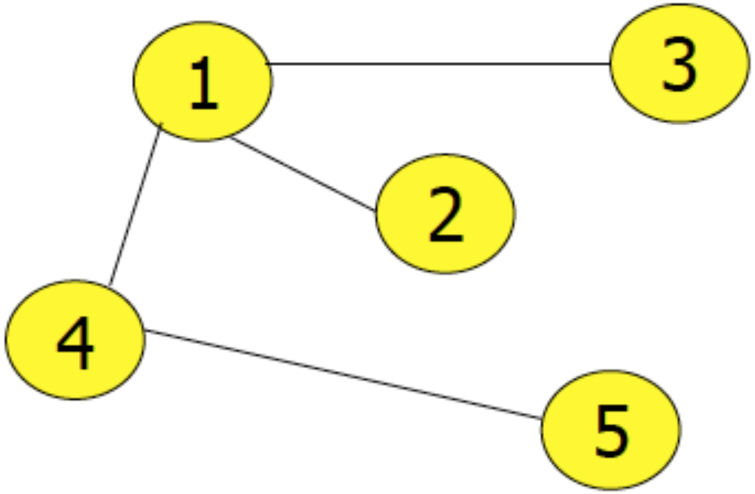


• Tree

- Definition: A tree is a connected undirected graph with no simple circuits.
- a connected graph with no cycles



Examples of a Tree



Tree

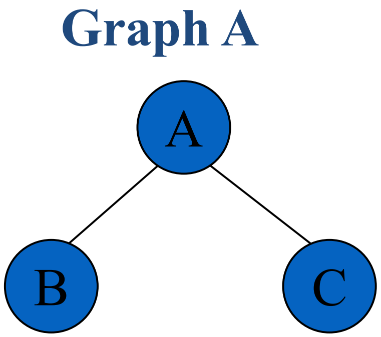
Subgraph

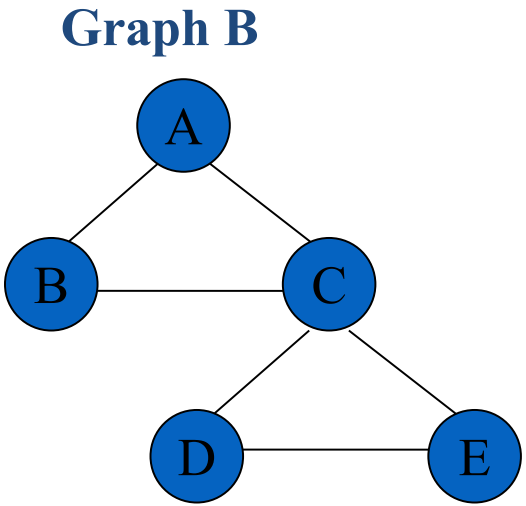
- Suppose that V(G) and E(G) denote the vertex and edge sets of a graph G. If H is graph with the properties:

V(H) ⊆V(G)
E(H) ⊆ E(G)

Every edge of E(H) has both its incident vertices in V(H) then H is called a **subgraph of G**

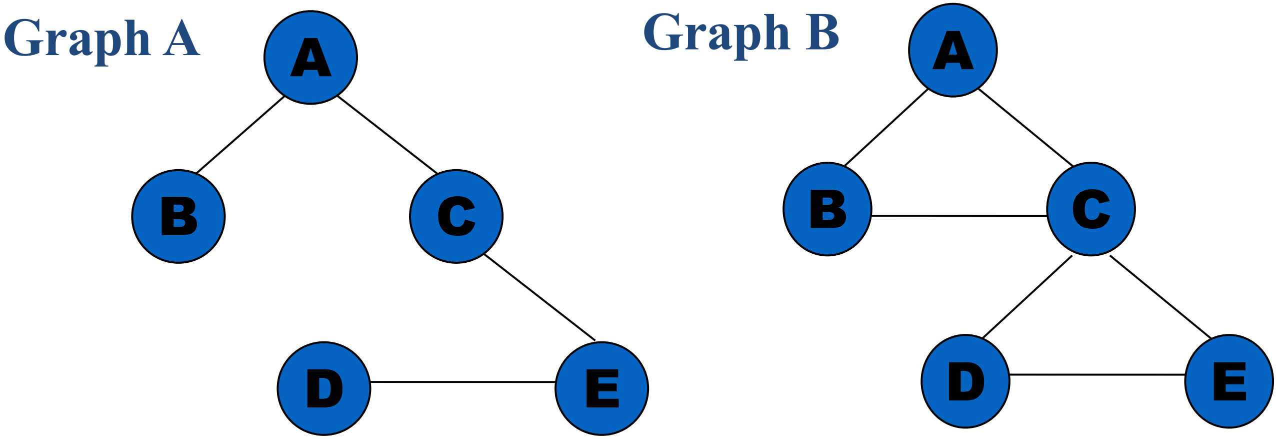
Subgraph





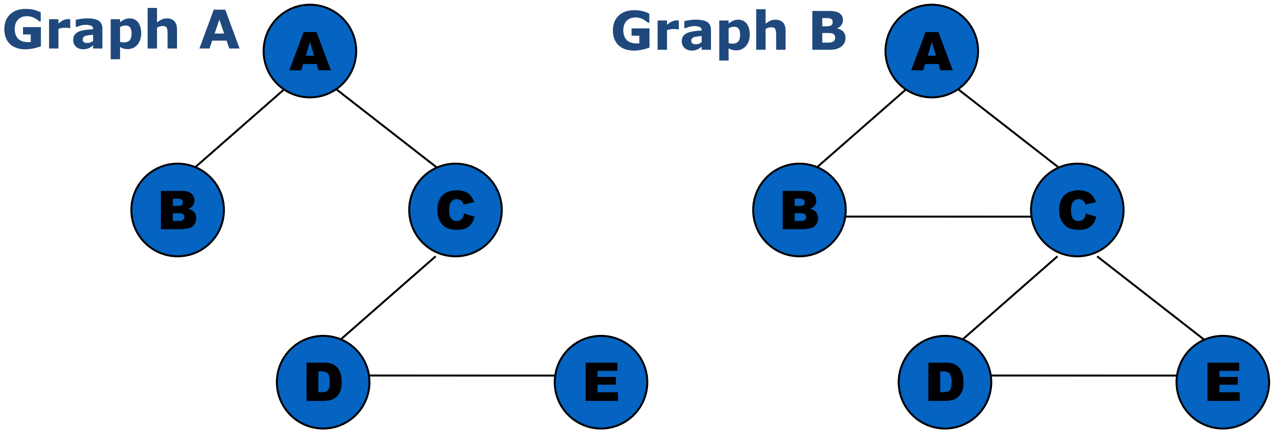
A is a subgraph of B

- If *V(H)=V(G)* then ***H*** is called a ***spanning subgraph*** of ***G***



A is a spanning subgraph of B

- If *H* is a tree, then *H* is called a **spanning tree**



A is a spanning tree of B

Spanning Trees

- A spanning tree of a graph is just a sub graph that contains all the vertices and is a tree.

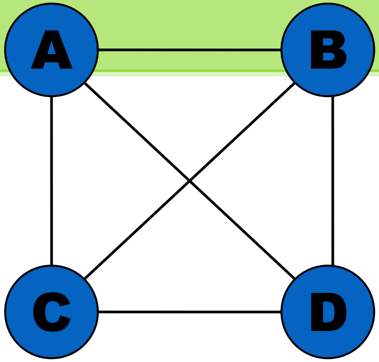
- A graph may have many spanning trees.

- for each graph G with n vertices , any spanning tree must has *n-1* edges, and *no cycle* on it.

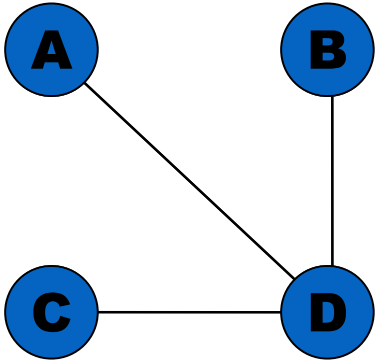
Spanning Tree properties:

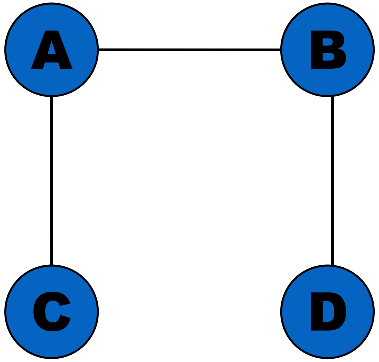
On a connected graph G=(V, E), a spanning tree *must be:*

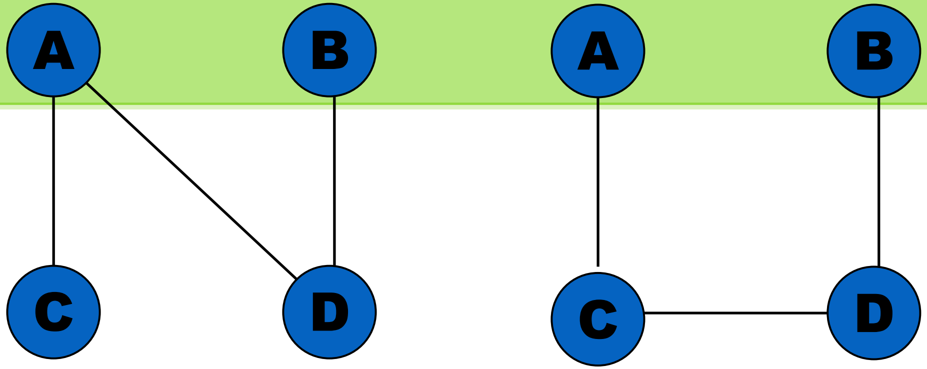
- a connected subgraph (contains all vertices of G)
- no cycle.
- is a tree (|E| = |V| - 1)



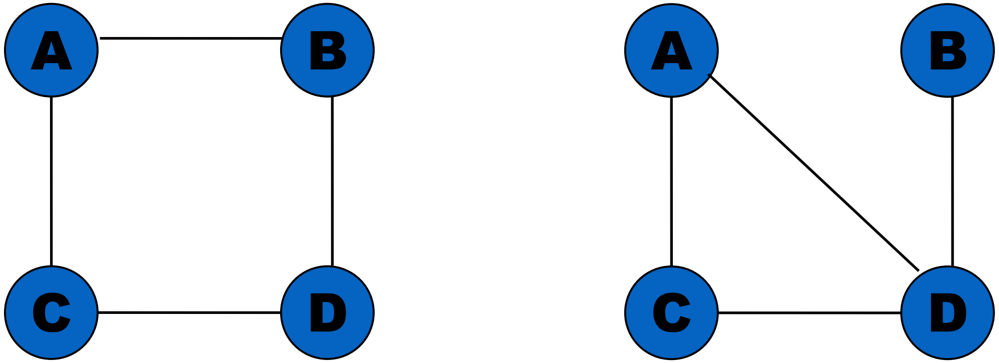
A connected undirected ***graph G***







Four of the ***spanning trees*** of the graph



The two ***are not*** spanning tree

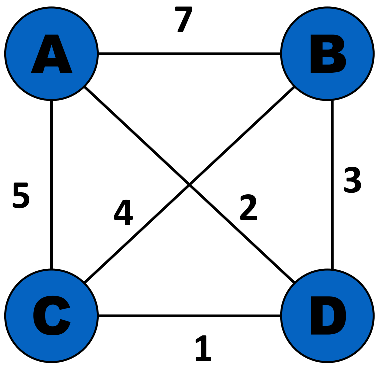
Minimum Spanning Tree

- The Minimum Spanning Tree for a given graph is the Spanning Tree of minimum cost for that graph .

- a minimum spanning tree (MST) is a spanning tree of minimum weight

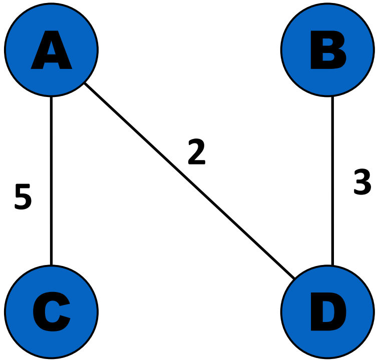
Note : we need to have spanning tree that connected to all its vertices but has less weights.

Example:

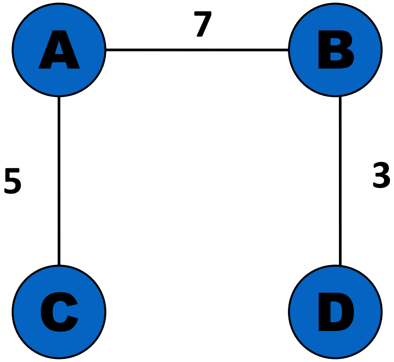


Complete Graph

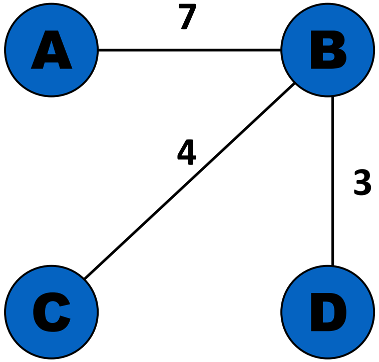
Note: Number of spanning tree of complete graph = n^{n-2}



Total Weight = 10



Total Weight = 15



Total Weight = 14

All These subgraphs are spanning tree, all its vertices are connected and there is no cycle. **But, they are not minimum spanning tree because the total weights are not the least total weight.**

MST Algorithms

- Minimum Spanning Tree
  - Kruskal’s Algorithm
  - Primm’s Algorithm

Applications

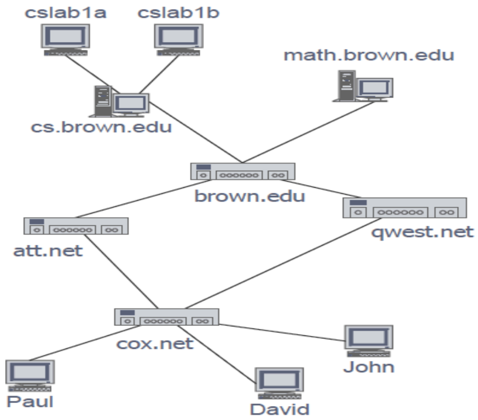
◆**Computer networks –**

- Local area network
- Internet

◆**Transportation networks**

- highway network
- flight network

◆**communication networks**



The End .